Bayesian Statistical Model Checking for Multi-agent Systems using HyperPCTL*

Spandan Das
Department of Computer Science
Kansas State University
Manhattan, Kansas
spandan@ksu.edu

Pavithra Prabhakar
Department of Computer Science
Kansas State University
Manhattan, USA
pprabhakar@ksu.edu

Abstract—In this paper, we present a Bayesian method for statistical model checking (SMC) of probabilistic hyperproperties specified in the logic HyperPCTL* on discrete-time Markov chains (DTMCs). While SMC of HyperPCTL* using sequential probability ratio test (SPRT) has been explored before, we develop an alternative SMC algorithm based on Bayesian hypothesis testing. In comparison to PCTL*, verifying HyperPCTL* formulae is complex owing to their simultaneous interpretation on multiple paths of the DTMC. In addition, extending the bottom-up model-checking algorithm of the non-probabilistic setting is not straight forward due to the fact that SMC does not return exact answers to the satisfiability problems of subformulae, instead, it only returns correct answers with high-confidence. We propose a recursive algorithm for SMC of HyperPCTL* based on a modified Bayes' test that factors in the uncertainty in the recursive satisfiability results. We have implemented our algorithm in a Python toolbox, HyProVer, and compared our approach with the SPRT based SMC. Our experimental evaluation demonstrates that our Bayesian SMC algorithm performs better both in terms of the verification time and the number of samples required to deduce satisfiability of a given HyperPCTL* formula.

Index Terms—Statistical Model Checking, Bayesian Hypothesis Testing, Probabilistic Hyperproperties

I. INTRODUCTION

Formal verification of software systems that interact with physical environments as in cyber-physical systems, has gained significant importance in recent years owing to the safety-criticality of these systems. Probabilistic and stochastic models have been proven to be useful tools in modeling the uncertainty inherent in the interactions with the environment. In particular, discrete-time Markov chains are a simple, yet widely applicable, class of probabilistic systems that capture uncertainties using transition probabilities. Correctness specifications of these systems need to incorporate the probabilistic aspects and are often captured using probabilistic logics. For example, uncertain behaviour of a car driver has been modeled as a discrete-time Markov chain (DTMC) [1], and several behavioral properties of the driver are encoded using the Probabilistic Computation Tree Logic (PCTL). Further, a robot performing random walk on a grid has been modeled using DTMC [2], and properties like probabilistic goal reaching have been encoded using Continuous Stochastic Logic (CSL).

Traditional logics focus on properties about single execution traces of a system. While several interesting properties can be captured using single-trace logics, they fall short in capturing interactive behaviors between multiple agents. Hyperproperties are a new class of properties [3], [4] that capture multi-trace behaviors. For example, consider two robots walking on an $n \times n$ grid. The property that the two robots never collide, can only be expressed by a formula that refers to the paths of both the robots simultaneously [5]. In addition, several security properties can be easily captured using hyper-logics such as information flow [6], [7], non-interference [4], [8] and observational determinism [8], [9].

In this paper, we focus on the problem of verifying discrete-time Markov chains with respect to probabilistic hyperproperties that allow us to specify constraints on joint probability of satisfaction of real-time behaviors by independent executions of a multi-agent system. We focus on discrete-time systems and properties, however, the ideas in the paper will be foundational toward verification of these systems and properties in continuous time. We consider robot navigation problems as our case studies. For example, consider a two robot navigation scenario on a grid, wherein we desire to ensure an upper bound on the collision probability. Such properties can be easily specified using probabilistic hyperproperty logics such as HyperPCTL*, which is an expressive hyperproperty logic that allows nesting of both temporal and probabilistic operators (see [10]). Thus, it has found its use in formal specification of probabilistic hyperproperties of cyber-physical and robotic systems [10]. Several quantitative extensions of information security properties have been captured using HyperPCTL* such as qualitative information flow [11], probabilistic non-interference [12] and differential privacy [13].

Probabilistic model-checkers such as PRISM [14] and STORM [15] verify probabilistic systems with respect to probabilistic properties. While probabilistic model-checking performs reasonably well for single trace properties, even in the non-probabilistic setting, model-checking hyperproperties is a challenging task [16]. Hence, light-weight verification methods based on sampling, such as statistical model-checking [10], [17] have been explored.

Statistical Model-Checking (SMC) [18]–[28] is an alternative sampling based method to verify satisfiability of the specification. More precisely, SMC algorithms collect a sample of execution paths from the system and determine the

satisfiability of the property based on a statistic computed using the satisfiability of these sample execution paths. SMC scales extremely well for complex systems, however, it does not provide exact inference, rather, incurs Type-I and Type-II errors which correspond to the probability that, the test concludes that the system violates the property when indeed it is satisfied, and the probability that, the test concludes that the system satisfies the property when actually it is violated, respectively. SMC for probabilistic hyperproperties has been discussed in [10], [17], where sequential probability ratio test (SPRT) has been used. These algorithms fall into the frequentist regime where the parameter is assumed to be fixed but unknown. In this paper, we explore an alternative approach based on Bayesian hypothesis testing which incorporates prior information about the parameters in the form of their distributions. We present a Bayesian SMC algorithm based on Bayes' test that takes as input Type-I and Type-II error bounds and provides corresponding guarantees on the inference of the satisfiability.

SMC of probabilistic hyperproperties specified in HyperPCTL* is challenging due to two reasons: HyperPCTL* in interpreted over multiple traces and the probabilistic operators are nested. To address the multi-trace setting, we present a multi-dimensional hypothesis testing and incorporate that within a recursive algorithm to tackle the nesting in probabilistic operators. Our broad approach to verifying a HyperPCTL* formula φ is a bottom up algorithm similar to classical CTL model-checking [29], [30], wherein we verify the bottom level probabilistic operators and work our way upwards using the results from the verification of the subformulae. Since, SMC only returns the correct answers with certain confidence, this error needs to be factored into the SMC procedures for top level formulae. We tackle this by designing a modified Bayes' test that uses a statistic computed from these approximately correct answers. While such a framework has been developed in the single trace setting [2], the main contribution of the paper is the careful incorporation of multi-dimensional Bayesian hypothesis testing into the recursive framework by establishing appropriate bounds on the confidence of lower level multi-dimensional SMC calls.

We have implemented our algorithm in a Python toolbox HyProVer and tested it on a series of robot navigation scenarios in the grid world setting (see Section V). Though the algorithm is written in a bottom-up fashion, we have implemented an efficient top-down approach, wherein lower level SMC calls are made only on a need-by basis, thereby avoiding an exhaustive computation of lower level SMC calls. We also implemented the SPRT approach for comparison. We observed that, for the same level of confidence, our approach takes fewer number of samples and less time to verify both non-nested and nested HyperPCTL* formulae compared to SPRT [10] (none of the case studies discussed in [10] involved a nested probabilistic formula). Also, our approach does not need the assumption that the true probability lies outside the *indifference region* (for non-nested formula) and thus, is more general than SPRT. Thus, our approach is more

practical for verification of general HyperPCTL* formulae. Bayesian approaches are often criticized for their need for prior information. We considered different choices for prior by considering various parameters for the Beta distribution and observed that priors effect the verification time in a minor manner. However, since SPRT does not use prior information, we used uniform prior while comparing Bayesian approach to SPRT, so that any parameter value is equally probable.

To summarize, the main contributions of this paper are:

- A first Bayesian statistical model-checking algorithm for HyperPCTL* that combines multi-dimensional Bayesian hypothesis testing and a recursive framework for error propagation of nested probabilistic operators.
- A novel top-down implementation that makes lower level SMC calls on-need and consists of additional bookkeeping to avoid redundant and unnecessary computation.
- Experimental evaluations and comparisons with existing approaches that demonstrate the scalability and benefits of our approach.

Organization: The rest of the paper is organized as follows. We discuss related work in Section II. Some basic definitions and notations are covered in Section III. We define the model checking problem formally in Section IV. The robot navigation system grid world and two properties of it are described in Section V. In Section VI, we discuss general and multi-dimensional hypotheses testing, as well as, Bayes' test, approximate Bayes' test and sequential probability ratio test (SPRT). Section VII explains the recursive Bayesian SMC algorithm and compares its approach with that of SPRT based SMC [10]. In Section VIII, we evaluate our algorithm on the case study discussed in Section V. Finally, we conclude in Section IX.

II. RELATED WORKS

Broadly, two classes of techniques have been explored for verification of probabilistic properties on probabilistic system models. Probabilistic model checking (PMC) techniques based on numerical methods that compute exact probability of satisfaction of a specification given in logics such as LTL [31], PCTL [32], [33], CSL [34], Bounded Temporal Logic (BTL) [35] and ω-regular languages [36] on stochastic models such as discrete-time Markov chain (DTMC) [31]–[33], [36], continuous time Markov chain (CTMC) [34], [35], Markov decision process (MDP) [33], [37] and ω-automaton [31] have been explored. However, these methods are generally model specific and involve solving a system of linear equations [33], [35] or complex properties of algebraic and transcendental number theory [34], which makes them computationally intensive.

On the other hand, statistical model checking (SMC) algorithms are based on random sampling of the probabilistic models and use statistical tests to provide (with certain confidence) inference on whether probability of a property lies within a certain range. Although SMC can only provide answers with certain amount of errors, it is much less computation intensive and thus used in a large number of real life applications [26], [38]. SMC has been used for verification of properties

specified in LTL [18], [19], Bounded LTL [27], Probabilistic Bounded LTL [28], PCTL [25] and CSL [2], [23], [24] where DTMC [2], [23], [25], CTMC [23], MDP [27], [39], semi-Markov process (SMP) [23], [24], generalised semi-Markov process (GSMP) [24] and various Stochastic Hybrid Systems (SHS) [18]–[20], [22], [28], [40] have been considered for the underlying probabilistic models. A brief discussion on state of the art SMC techniques (e.g., sampling and testing methods) can be found from several survey papers [41], [42]. For example, sampling techniques like Monte-Carlo [19] and perfect simulation [21] have been used to sample various probabilistic models, and statistical tests like Bayesian [2], [28], importance sampling [43], acceptance sampling [24] have been used to gather inference about their probabilistic properties. Detailed study on various statistical tests and their applications has been performed [44], [45]. Tools like Apmc [46], PRISM [14], STORM [15] have been developed that can automatically verify probabilistic properties on probabilistic models.

Encoding multi-trace properties (known as hyperproperties) using logics like LTL, PCTL, CSL is not straightforward as they mainly capture properties about individual traces of the underlying model. Logics like Bounded HyperLTL [4], HyperLTL [3], HyperCTL* [3] were defined for specification of hyperproperties. More recently, HyperPSTL [17] and HyperPCTL* [10] have been defined to encode probabilistic hyperproperties. Model checking of probabilistic hyperproperties is a relatively new area of research; one recent work explores a statistical model-checking algorithm based on Clopper-Pearson interval method [17] and sequential probability ratio test (SPRT) [10]. In this paper, we explore a Bayesian approach for verification of probabilistic hyperproperties. To the best of our knowledge, this is the first statistical model-checking algorithm for probabilistic hyperproperties based on a Bayesian approach.

III. PRELIMINARIES

Let us define some basic terms and notations that are used in the paper. Given a sequence $\sigma = s_1, s_2, \ldots$, $\sigma[i]$ denotes i^{th} element of the sequence σ, that is, s_i, and $|\sigma|$ denotes length of the sequence σ. For an n-tuple $\bar{X} = (X_1, \ldots, X_n)$, $\bar{X}[i]$ denotes the ith element of $\bar{X}$, that is, X_i. For an n-tuple $\bar{X}$, infinite norm is denoted by $\|\bar{X}\|_\infty$ where, $\|\bar{X}\|_\infty = \max_{i=1}^n \bar{X}[i]$. For any natural number n, $[n]$ denotes the set of natural numbers $\{1, \ldots, n\}$.

Recall that, given a continuous random variable X, the function $F : \mathbb{R} \to [0,1]$ defined as $F(x) = P(X \leqslant x)$ is called the *cumulative distribution (cdf)* of X and the function $f : \mathbb{R} \to \mathbb{R}$ defined as $f(x) = \frac{d}{dx} F(x)$ is called the *probability density (pdf)* of X.

For any $x \in \mathbb{R}^n$, the ϵ-ball around x with respect to the infinite norm is defined as $B_\epsilon(x) := \{y \in \mathbb{R}^n : \|x - y\|_\infty \leqslant \epsilon\}$. For a set $A \subseteq \mathbb{R}^n$, the boundary of A (denoted $\partial(A)$) is defined as the set of all $x \in \mathbb{R}^n$ such that $B_\epsilon(x) \cap A \neq \varnothing$ and $B_\epsilon(x) \cap A^c \neq \varnothing$, for all $\epsilon > 0$.

IV. HYPERPCTL* VERIFICATION PROBLEM

In this section, we formalize the model checking problem of a system given as a Discrete-time Markov Chain with respect to correctness criterion specified in the logic HyperPCTL* [10] which specifies hyperproperties.

A. Discrete-time Markov Chain

A discrete-time Markov chain is a structure that consists of a set of states along with transitions that specify the probability of the next state of the system given the current state.

Definition 1 (Discrete-time Markov Chain): A discrete-time Markov Chain (DTMC) is a tuple $\mathcal{M} = (S, R, AP, L)$ where,

- S is the finite set of states;
- $R : S \times S \to [0,1]$ is the *transition probability function* such that for any $s \in S$, $\sum_{s' \in S} R(s, s') = 1$;
- AP is the set of atomic propositions; and
- $L : S \to 2^{AP}$ is the labeling function that associates a set of atomic proposition to each state.

A path (trace) of a DTMC $\mathcal{M}$ is a sequence of states $\sigma = s_1, s_2, \ldots$ such that for all $i < |\sigma|$, $R(s_i, s_{i+1}) > 0$. Paths$(\mathcal{M})$ denotes the set of all infinite paths and Paths$_{fin}(\mathcal{M})$ denotes the set of all finite paths of a DTMC $\mathcal{M}$.

B. HyperPCTL* Logic

Next, we define the syntax and semantics of HyperPCTL* which is a logic for specifying probabilistic hyperproperties and is interpreted on multiple traces from a DTMC.

1) Syntax of HyperPCTL:* Let us fix a set of atomic propositions AP and a (possibly infinite) set of path variables Π. HyperPCTL* formulae over AP and Π are defined by the following grammar:

$$\phi := a^\pi \mid \neg\phi \mid \phi \land \phi \mid \bigcirc\phi \mid \phi U^{\leqslant k}\phi$$
$$\mid \mathbb{P}_D(Pr^{\bar{\pi}}(\phi), \ldots, Pr^{\bar{\pi}}(\phi)), \quad (1)$$

where $a \in AP$ is an atomic proposition and $\pi \in \Pi$ is a path variable. $\bigcirc$ and $U^{\leqslant k}$ are the 'next' and 'until' operators respectively, where $k \in \mathbb{N} \cup \{\infty\}$ is the time bound. For this work, we will assume $k \in \mathbb{N}$, i.e., we will only consider 'until' operators with finite time bound. $Pr^{\bar{\pi}}$ is the probability operator for a tuple of path variables $\bar{\pi} = (\pi_1, \ldots, \pi_n)$, where $n \in \mathbb{N}$ and $\pi_i \in \Pi$ for all $i \in [n]$. $\mathbb{P}_D(x_1, \ldots, x_n)$ is an n-ary predicate function which is satisfied iff $(x_1, \ldots, x_n) \in D \subseteq \mathbb{R}^n$.

Additional logic operators are derived as usual: $\top \equiv a^\pi \land \neg a^\pi$, $\phi \land \phi' \equiv \neg(\neg\phi \lor \neg\phi')$, $\phi \Rightarrow \phi' \equiv \neg\phi \lor \phi'$, $\diamondsuit^{\leqslant k}\phi \equiv \top U^{\leqslant k}\phi$, and $\square^{\leqslant k}\phi \equiv \neg\diamondsuit^{\leqslant k}\neg\phi$. We represent a 1-tuple by its element, i.e., $Pr^{(\pi)}$ is written as Pr^π.

Remark 2: In general, HyperPCTL* formulae are defined by the grammar:

$$\phi := a^\pi \mid \phi^\pi \mid \neg\phi \mid \phi \land \phi \mid \bigcirc\phi \mid \phi U^{\leqslant k}\phi \mid \rho \bowtie \rho$$
$$\rho := f(\rho, \ldots, \rho) \mid Pr^{\bar{\pi}}(\phi) \mid Pr^{\bar{\pi}}(\rho),$$

where $\bowtie \in \{>, <, \geqslant, \leqslant, =\}$. Note that, a formula of the form $\rho_1 \bowtie \rho_2$ can be equivalently depicted as $\mathbb{P}_D(\rho_1, \rho_2)$, where $D = \{(x_1, x_2) \mid x_1 - x_2 \bowtie 0\}$. Similarly, other ρ formulae

can also be expressed by the grammar in Equation 1 defined above, i.e., the grammar in Equation 1 is as expressive as the general HyperPCTL* grammar.

2) Semantics of HyperPCTL*: A HyperPCTL* formula over *AP* and Π is interpreted on the pair $(\mathcal{M}, V)$, where $\mathcal{M}$ is a DTMC with propositions *AP*, and $V{:}\Pi \to \text{Paths}(\mathcal{M})$ is a mapping from Π to Paths($\mathcal{M}$). We say V is a *path assignment*. $[\![\phi]\!]_V$ denotes instantiation of the mapping V on the formula ϕ, i.e., each $\pi \in \Pi$ that appears in ϕ is replaced by the path $V(\pi) \in \text{Paths}(\mathcal{M})$. Let $\bar{\pi}$ and $\bar{\sigma}$ be sequences of path variables and paths of $\mathcal{M}$ respectively such that $|\bar{\pi}| = |\bar{\sigma}|$. $V[\bar{\pi} \to \bar{\sigma}]$ denotes revision of the mapping V where $\bar{\pi}[i]$ is remapped to $\bar{\sigma}[i]$ for all $i \in [|\bar{\pi}|]$. $V^{(i)}$ denotes the i-shift of V, i.e., $V^{(i)}(\pi)$ is the path $(V(\pi)[i], V(\pi)[i+1], \dots)$. Paths$(V, \pi)$ denotes the set of all paths $\sigma \in \text{Paths}(\mathcal{M})$ such that $\sigma[0] = V(\pi)[0]$ and Paths$(V, \bar{\pi})$ denotes the set of all path sequences $\bar{\sigma} \in (\text{Paths}(\mathcal{M}))^*$ such that $(\bar{\sigma}[i])[0] = V(\bar{\pi}[i])[0]$ for all i. The semantics of HyperPCTL* is described as follows,

$$(\mathcal{M}, V) \models a^\pi \quad \text{iff} \quad a \in L(V(\pi)(0))$$
$$(\mathcal{M}, V) \models \neg\phi \quad \text{iff} \quad (\mathcal{M}, V) \not\models \phi$$
$$(\mathcal{M}, V) \models \phi_1 \wedge \phi_2 \quad \text{iff} \quad (\mathcal{M}, V) \models \phi_1 \text{ and } (\mathcal{M}, V) \models \phi_2$$
$$(\mathcal{M}, V) \models \bigcirc\phi \quad \text{iff} \quad (\mathcal{M}, V^{(1)}) \models \phi$$
$$(\mathcal{M}, V) \models \phi_1 U^{\leq k}\phi_2 \quad \text{iff} \quad \exists i \leq k \text{ such that}$$
$$((\mathcal{M}, V^{(i)}) \models \phi_2) \wedge (\forall j < i, (\mathcal{M}, V^{(j)}) \models \phi_1)$$
$$(\mathcal{M}, V) \models \mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \dots, Pr^{\bar{\pi}_n}(\phi_n)) \quad \text{iff}$$
$$([\![Pr^{\bar{\pi}_1}(\phi_1)]\!]_V, \dots, [\![Pr^{\bar{\pi}_n}(\phi_n)]\!]_V) \in D$$
$$[\![Pr^{\bar{\pi}}(\phi)]\!]_V = P\{\bar{\sigma} \in \text{Paths}(V, \bar{\pi}) \mid (\mathcal{M}, V[\bar{\pi} \to \bar{\sigma}]) \models \phi\}$$

The expression $P\{\bar{\sigma} \in \text{Paths}(V, \bar{\pi}) \mid (\mathcal{M}, V[\bar{\pi} \to \bar{\sigma}]) \models \phi\}$ denotes the probability of satisfaction of ϕ on the set of path sequences $\{\bar{\sigma} : |\bar{\sigma}| = |\bar{\pi}| \text{ and } (\bar{\sigma}[i])[0] = V(\bar{\pi}[i])[0] \; \forall i\}$.

C. The Model Checking Problem

Let $\mathcal{M}$ be a DTMC and ϕ be a HyperPCTL* formula. Let Π_ϕ denote the finite set of path variables used to define ϕ and $V : \Pi \to \text{Paths}(\mathcal{M})$ be a path assignment. The model checking problem is to decide whether $\mathcal{M}, V \models \phi$ given a DTMC $\mathcal{M}$ and path assignment V. Note that, since we encounter path variables that appear in ϕ only and all linear time operators are bounded, it is enough to provide a path assignment $V : \Pi_\phi \to \text{Paths}_{\textit{fin}}(\mathcal{M})$, as satisfaction of a formula depends only on finite prefixes of assignments.

V. CASE STUDY: GRID WORLD

In this section, we describe a robot navigation scenario on a grid world and discuss two desirable properties. Consider an $n \times n$ grid where N robots are performing 2 dimensional random walks. In other words, each robot can move to a cell to its left, right, above or below only (if possible) with non zero probability. In Figure 1, a 4×4 grid with two robots has been shown. Each robot has a particular goal which is one of the cells of the grid. In Figure 1, the goal for the first robot (red) has been marked by g_1 and goal for the second
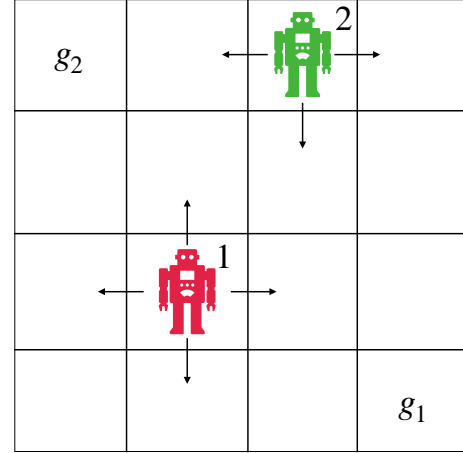

Fig. 1: Robot Motion in Grid World

robot (green) has been marked by g_2. The first property we would like to verify is that the robots do not collide with each other within a finite number of steps with high probability. The second property we would like to verify is that the robots reach their respective goals within a finite number of steps with high probability while avoiding collision with each other (with high probability).

A. DTMC Modeling

The robot navigation system can be aptly modeled using a discrete-time Markov chain (DTMC), where, a state can uniquely represent the position and identity of a robot and the atomic propositions represent individual cells properties. In other words, a state is of the form q_{ijk} where, (i, j) denotes the position and k denotes the identity of the robot. Thus, we have the set of states $S = \{q_{ijk} \mid i, j \in [n] \text{ and } k \in [N]\}$. Each state q_{ijk} is marked with the atomic proposition a_{ij}, which denotes the position corresponding to that state, i.e., $a_{ij} \in L(q_{ijk})$ for all k. Also, q_{ijk} is marked with the proposition g_k if (i, j) is the goal of robot k, i.e., $g_k \in L(q_{ijk})$ iff (i, j) is the goal of robot k. This gives us the set of all atomic propositions as $AP = \{a_{ij} \mid i, j \in [n]\} \cup \{g_i \mid i \in [N]\}$ and the labeling function L. The transition function R ensures that a robot can move from cell (i, j) to cell (i', j') only if they share a common boundary, i.e., $R(q_{ijk}, q_{i'j'k'}) > 0$ iff cells (i, j) and (i', j') share a common boundary and $k = k'$. This probability might vary for different robots. Incorporating k as an index of the states ensures that the probability $R(q_{ijk}, q_{i'j'k})$ can be uniquely defined for each robot $k \in [N]$.

B. Collision Avoidance

A collision between two robots happens if they reach the same cell at the same point of time. Avoiding collision is a desirable property. In other words, assuming π_1 is the path followed by the first robot and π_2 is the path followed by the second robot on the DTMC, we would like that a_{ij} not in both $\pi_1[k]$ and $\pi_2[k]$ for all $k \in \mathbb{N}$. Since we are only considering

bounded formulae, we assume $k \leqslant K$ for some $K \in \mathbb{N}$. Thus an interesting property of the grid world will be to check if the probability that a_{ij} in both $\pi_1[k]$ and $\pi_2[k]$ for some $i, j \in [n]$ and for any $k \leqslant K$ is smaller than a certain threshold θ. This can be represented by the HyperPCTL* formula ψ_{ca}:

$$\mathbb{P}_{[0,\theta]} \left(Pr^{(\pi_1, \pi_2)} \left[\diamondsuit^{\leqslant K} \left(\bigvee_{i,j \in [n]} (a_{ij}^{\pi_1} \wedge a_{ij}^{\pi_2}) \right) \right] \right). \quad (2)$$

C. Collision Free Goal Reaching

Another desirable property of the grid world is that a robot reaches its goal within some finite number of steps with high probability while avoiding collision with other robots with high probability. For an $n \times n$ grid with two robots, this property can be aptly represented for the first robot using the nested HyperPCTL* formula ψ_{goal}:

$$\mathbb{P}_{[\theta_1, 1]} \left(Pr^{\pi_1} \left[(\psi_{nocol}) U^{\leqslant K} (g_1^{\pi_1}) \right] \right) \text{ where,} \quad (3)$$

$$\psi_{nocol} = \mathbb{P}_{[\theta_2, 1]} \left(Pr^{\pi_2} \left[\neg \left(\bigvee_{i,j \in [n]} (a_{ij}^{\pi_1} \wedge a_{ij}^{\pi_2}) \right) \right] \right).$$

Observe that, this formula represents the property that the first robot (following the path π_1) reaches its goal within K steps with probability at least θ_1 while avoiding collision with the second robot (following the path π_2) with probability at least θ_2. We can state this property for the second robot as well in a similar manner by simply interchanging the positions of π_1 and π_2 and replacing g_1 by g_2.

VI. HYPOTHESIS TESTING

The objective of a hypothesis test is to make some inference on the parameter(s) of a probability distribution using some statistical tests. More precisely, consider a random variable Y with pdf f that depends on some parameter Θ. The goal of hypothesis testing is to determine whether Θ lies above or below a certain value θ_0 known as the threshold. Thus we obtain two hypotheses:

$$H_0 : \Theta \geqslant \theta_0 \text{ vs } H_1 : \Theta < \theta_0.$$

To determine whether H_0 or H_1, referred to as the null and alternate hypothesis, respectively, is true, a hypothesis test consists of sampling the random variable Y to obtain data y, which is a sequence of values of Y observed while sampling, computing a statistic $T(y)$, where T is a function that maps y to some real number $T(y)$, and accepting or rejecting H_0 (with probability greater than a desired threshold) based on the value of $T(y)$. The statistic T should be chosen such that the corresponding test provides the correct inference with high probability. There are two types of errors associated with a hypothesis test: the probability of rejecting H_0 when H_0 is true, referred to as the Type-I error, and the probability of accepting H_0 when H_0 is false, referred to as the Type-II error. It is desirable that Type-I and Type-II errors are bounded above by some small positive quantities $\alpha, \beta \in (0, 1)$ (respectively).

A. Multidimensional Hypothesis Testing

Our verification problem translates to a hypothesis testing problem on a vector of random variables (random vector) each with its own parameter. Here, we set up the multi-dimensional hypothesis testing problem that involves a multi-dimensional parameter corresponding to a multi-dimensional random vector. Let $\bar{Y} = (Y_1, \ldots, Y_n)$ be an n-dimensional random vector formed by independent random variables $Y_1, \cdots, Y_n$ with parameter vector $\bar{\Theta} = (\Theta_1, \ldots, \Theta_n)$, that is, Y_i has parameter Θ_i. The goal of n-dimensional hypothesis testing is to decide whether $\bar{\Theta}$ is in $D \subseteq \mathbb{R}^n$ or not. Hence, our hypothesis testing problem is to decide between the following two hypotheses:

$$H_0 : \bar{\Theta} \in D \quad \text{vs} \quad H_1 : \bar{\Theta} \in D^c. \quad (4)$$

We obtain a sequence of N samples $\bar{y} = (\mathbf{y_1}, \ldots, \mathbf{y_N})$ from the n-dimensional random vector $\bar{Y} = (Y_1, \ldots, Y_n)$, that is, each $\mathbf{y_i}$ is a tuple of values $(y_{i1}, \ldots, y_{in})$ where y_{ij} is sampled from the random variable Y_j. We compute a statistic T of $\bar{y}$, and compare it with a value that depends on D to determine whether H_0 should be accepted or rejected.

B. Bayesian Hypothesis Testing

Bayesian hypothesis testing is a hypothesis testing framework wherein we assume some knowledge about the parameters in the form of prior distributions. Bayesian methods often perform better in terms of the inference since they factor in additional information about the parameters. Hence, we are given a random vector $\bar{Y}$ of i.i.d. random variables along with the parameter vector $\bar{\Theta}$ and a joint pdf $f_{\bar{\Theta}}$ that is known. We compute a statistic known as the Bayes' factor, denoted $\mathbb{B}_{\bar{Y}}(\bar{y}, D)$, given by the ratio of the probability that the data $\bar{y}$ is observed given H_0 is true (denoted $P(\bar{y} \mid H_0)$) to the probability that $\bar{y}$ is observed given H_1 is true (denoted $P(\bar{y} \mid H_1)$). In other words,

$$\begin{aligned} \mathbb{B}_{\bar{Y}}(\bar{y}, D) &= \frac{P(\bar{y} \mid H_0)}{P(\bar{y} \mid H_1)} \\ &= \frac{\int_{\theta \in D} P_{\bar{Y}|\Theta}(\bar{y} \mid \theta) f_{\bar{\Theta}}(\theta) d\theta}{\int_{\theta \in D^c} P_{\bar{Y}|\Theta}(\bar{y} \mid \theta) f_{\bar{\Theta}}(\theta) d\theta} \cdot \frac{P_1}{P_0} \end{aligned} \quad (5)$$

where $P_0 = \int_{\theta \in D} f_{\bar{\Theta}}(\theta) d\theta$ and $P_1 = \int_{\theta \in D^c} f_{\bar{\Theta}}(\theta) d\theta$.

The following theorem [2], [28] provides us a way to perform hypothesis testing using Bayes' factor.

Theorem 3: [Bayes' test] Let $\bar{Y}$ be a discrete random vector with parameter $\bar{\Theta}$ and corresponding joint pdf $f_{\bar{\Theta}}$. Let,

$$H_0 : \bar{\Theta} \in D \text{ vs } H_1 : \bar{\Theta} \in D^c,$$

be the null and alternative hypotheses respectively. Consider the test that

- accepts H_0 when $\mathbb{B}_{\bar{Y}}(\bar{y}, D) \geqslant \frac{1}{\beta}$ and
- rejects H_0 when $\mathbb{B}_{\bar{Y}}(\bar{y}, D) \leqslant \alpha$.

Then α and β are the upper bounds of Type-I and Type-II errors, respectively.

In the sequel, the Y_i in our hypothesis test will be a Bernoulli random variable with parameter $\Theta_i \in (0, 1)$, denoted $Y_i \sim \mathcal{B}(\Theta_i)$. We consider Beta distributions, denoted

$Beta(a, b)$ with parameters a, b, for the prior, since they are defined on the interval $(0, 1)$ and can represent several important distributions for suitable choices of a and b. For example, the uniform distribution on $[0, 1]$, that is, $U[0, 1]$, can be represented by $Beta(1, 1)$.

C. Approximate Bayesian Hypothesis Testing

We will encounter a situation while designing the Bayesian SMC where we cannot obtain samples from Y_i itself, but can obtain samples from a distribution close to Y_i. Hence, we present a Bayesian hypothesis test for the parameters of $\bar{\mathbf{Y}}$ using samples from these approximate distributions. Consider the case when sampling directly from the n-dimensional random vector $\bar{\mathbf{Y}} = (Y_1, \ldots, Y_n)$ with parameter $\bar{\Theta}$ is not possible. Suppose instead, we can sample from the n-dimensional random vector $\bar{\mathbf{Z}} = (Z_1, \ldots, Z_n)$ with parameter vector $\bar{\Theta}' = (\Theta'_1, \cdots, \Theta'_n)$ with the following property:

$$P(Z_i = 0 \mid Y_i = 1) \leqslant \alpha'_i \quad \text{and}$$
$$P(Z_i = 1 \mid Y_i = 0) \leqslant \beta'_i. \tag{6}$$

for all $i \in [n]$, where $\alpha'_i, \beta'_i \in (0, 1)$ are small positive quantities. In other words, the distributions of $\bar{\mathbf{Y}}$ and $\bar{\mathbf{Z}}$ are "close".

Given this relation between random variables Y_i and Z_i for each i, the following proposition (a direct corollary of Proposition 1 in [2]) bounds the distance (associated with the infinite norm) between $\bar{\Theta}$ and $\bar{\Theta}'$.

Proposition 4: Let $\bar{\mathbf{Y}}$ and $\bar{\mathbf{Z}}$ be n-dimensional random vectors consisting of Bernoulli random variables, with parameters $\bar{\Theta}$ and $\bar{\Theta}'$ respectively. In other words, $\bar{\mathbf{Y}}[i] \sim \mathcal{B}(\bar{\Theta}[i])$ and $\bar{\mathbf{Z}}[i] \sim \mathcal{B}(\bar{\Theta}'[i])$ for all $i \in [n]$. Suppose Equation 6 holds for each $i \in [n]$, i.e.,

$$P(\bar{\mathbf{Z}}[i] = 0 \mid \bar{\mathbf{Y}}[i] = 1) \leqslant \alpha'_i \quad \text{and}$$
$$P(\bar{\mathbf{Z}}[i] = 1 \mid \bar{\mathbf{Y}}[i] = 0) \leqslant \beta'_i, \quad \forall i \in [n].$$

Then, $\|\bar{\Theta} - \bar{\Theta}'\|_\infty \leqslant \delta$ where, $\delta = \max_{i=1}^n \{\max\{\alpha'_i, \beta'_i\}\}$.
Proof. From Proposition 1 in [2] we have,

$$(1 - \alpha'_i)\bar{\Theta}[i] \leqslant \bar{\Theta}'[i]$$
$$\Rightarrow (\bar{\Theta}[i] - \bar{\Theta}'[i]) \leqslant \alpha'_i \cdot \bar{\Theta}[i] \leqslant \alpha'_i,$$

since $\Theta_i \leqslant 1$. Similarly, we also have,

$$\bar{\Theta}'[i] \leqslant \bar{\Theta}[i] + \beta'_i(1 - \bar{\Theta}[i])$$
$$\Rightarrow (\bar{\Theta}'[i] - \bar{\Theta}[i]) \leqslant \beta'_i(1 - \bar{\Theta}[i]) \leqslant \beta'_i,$$

since $(1 - \bar{\Theta}[i]) \leqslant 1$. Hence,

$$|\bar{\Theta}[i] - \bar{\Theta}'[i]| \leqslant \max\{(\bar{\Theta}'[i] - \bar{\Theta}[i]), (\bar{\Theta}[i] - \bar{\Theta}'[i])\}$$
$$\leqslant \max\{\alpha'_i, \beta'_i\}$$

for all $i \in [n]$. Thus, $\|\bar{\Theta} - \bar{\Theta}'\|_\infty \leqslant \delta$ where $\delta = \max_{i=1}^n \{\max\{\alpha'_i, \beta'_i\}\}$. $\square$

Our next goal is to device a test to deduce whether $H_0 : \bar{\Theta} \in D$ or $H_1 : \bar{\Theta} \in D^c$ is satisfied using samples from $\bar{\mathbf{Z}} = (Z_1, \ldots, Z_n)$. To this end, let us first define ϵ-expansion and ϵ-reduction of a set A as, $A^+_\epsilon = \{x \in \Omega \mid B_\epsilon(x) \cap A \neq \varnothing\}$

and $A^-_\epsilon = \Omega \backslash (A^c)^+_\epsilon$ respectively, where Ω is the sample space for $\bar{\Theta}$. In other words, A^+_ϵ is the set of all $x \in \Omega$ such that, the ϵ-ball around x has non-null intersection with the set A. Similarly, A^-_ϵ is the complement set of all $x \in \Omega$ such that, the ϵ-ball around x has non-null intersection with complement of A. Provided $P(D) \in (0, 1)$, we have the following theorem which provides us a test for $\bar{\Theta}$, parameter of $\bar{\mathbf{Y}}$, using samples from $\bar{\mathbf{Z}}$.

Theorem 5: [Approximate Bayes' test] Let $\bar{\mathbf{Y}}$ and $\bar{\mathbf{Z}}$ be discrete random vectors ($\bar{\mathbf{Y}}[i], \bar{\mathbf{Z}}[i]$ are Bernoulli random variables for all i) of same dimension with parameters $\bar{\Theta}$ and $\bar{\Theta}'$ respectively. Also, let $\|\bar{\Theta} - \bar{\Theta}'\|_\infty \leqslant \delta$. Let,

$$H_0 : \bar{\Theta} \in D \text{ vs } H_1 : \bar{\Theta} \in D^c$$

be the null and alternate hypotheses respectively. Consider the test that

- accepts H_0 when $\mathbb{B}_{\bar{\mathbf{Z}}}(\bar{\mathbf{z}}, D^-_\delta) \geqslant \frac{1}{\beta \cdot r_2}$ and
- rejects H_0 when $\mathbb{B}_{\bar{\mathbf{Z}}}(\bar{\mathbf{z}}, D^+_\delta) \leqslant \alpha \cdot r_1$,

where r_1 and r_2 are constants defined as,

$$r_1 = \frac{P(\bar{\Theta} \in D)}{P(\bar{\Theta} \in D^+_{2\delta})} \quad \text{and} \quad r_2 = \frac{P(\bar{\Theta} \in D^c)}{P(\bar{\Theta} \in (D^c)^+_{2\delta})}$$

Then, α and β are the upper bounds of Type-I and Type-II errors respectively.
Proof. Let us show that α is the Type-I error bound for the hypothesis test, that is, we show that $P(\{\text{reject } H_0\} \mid H_0) \leqslant \alpha$.

Note that, from Proposition 4, $\bar{\Theta}(\omega) \in D$ implies $\bar{\Theta}'(\omega) \in D^+_\delta$ and, $\bar{\Theta}'(\omega) \in D^+_\delta$ implies $\bar{\Theta}(\omega) \in D^+_{2\delta}$, as $\|\bar{\Theta} - \bar{\Theta}'\|_\infty \leqslant \delta$. Hence, $\{\omega \mid \bar{\Theta}(\omega) \in D\} \subseteq \{\omega \mid \bar{\Theta}'(\omega) \in D^+_\delta\} \subseteq \{\omega \mid \bar{\Theta}(\omega) \in D^+_{2\delta}\}$.

$$P(\{\text{reject } H_0\} \mid H_0) = P\left(\mathbb{B}_{\bar{\mathbf{Z}}}\left(\bar{\mathbf{z}}, D^+_\delta\right) \leqslant \alpha \cdot r_1 \mid \bar{\Theta} \in D\right)$$
$$= \frac{P\left(\mathbb{B}_{\bar{\mathbf{Z}}}\left(\bar{\mathbf{z}}, D^+_\delta\right) \leqslant \alpha \cdot r_1, \bar{\Theta} \in D\right)}{P(\bar{\Theta} \in D)}$$
$$\leqslant \frac{P\left(\mathbb{B}_{\bar{\mathbf{Z}}}\left(\bar{\mathbf{z}}, D^+_\delta\right) \leqslant \alpha \cdot r_1, \bar{\Theta}' \in D^+_\delta\right)}{P(\bar{\Theta} \in D)}$$
$$= \frac{P\left(\mathbb{B}_{\bar{\mathbf{Z}}}\left(\bar{\mathbf{z}}, D^+_\delta\right) \leqslant \alpha \cdot r_1, \bar{\Theta}' \in D^+_\delta\right)}{P\left(\bar{\Theta}' \in D^+_\delta\right)} \cdot \frac{P(\bar{\Theta}' \in D^+_\delta)}{P\left(\bar{\Theta} \in D\right)}$$
$$\leqslant P\left(\mathbb{B}_{\bar{\mathbf{Z}}}\left(\bar{\mathbf{z}}, D^+_\delta\right) \leqslant \alpha \cdot r_1 \mid \bar{\Theta}' \in D^+_\delta\right) \frac{P(\bar{\Theta} \in D^+_{2\delta})}{P\left(\bar{\Theta} \in D\right)}$$
$$= \frac{P\left(\mathbb{B}_{\bar{\mathbf{Z}}}\left(\bar{\mathbf{z}}, D^+_\delta\right) \leqslant \alpha \cdot r_1 \mid \bar{\Theta}' \in D^+_\delta\right)}{r_1}$$
$$\leqslant \frac{\alpha \cdot r_1}{r_1} \cdot P\left(\bar{\mathbf{z}} \mid \bar{\Theta}' \in (D^+_\delta)^c\right)$$
$$\text{[since } \mathbb{B}_{\bar{\mathbf{Z}}}(\bar{\mathbf{z}}, D^+_\delta) \leqslant r_1\alpha \text{ iff } P\left(\bar{\mathbf{z}} \mid \bar{\Theta}' \in D^+_\delta\right)$$
$$\leqslant r_1\alpha \cdot P\left(\bar{\mathbf{z}} \mid \bar{\Theta}' \in (D^+_\delta)^c\right)]$$
$$\leqslant \alpha \quad \text{[since } P\left(\bar{\mathbf{z}} \mid \bar{\Theta}' \in (D^+_\delta)^c\right) \leqslant 1]$$

Similarly, we can show that β is the Type-II error bound for the hypothesis test, i.e., $P(\{\text{accept } H_0\} \mid H_1) \leqslant \beta$.

$\square$

Remark 6: Note that, Theorem 5 (approximate Bayes' test) essentially tests

$$H_0 : \bar{\mathbf{\Theta}}' \in D_\delta^- \text{ vs } H_1 : \bar{\mathbf{\Theta}}' \in (D_\delta^+)^c.$$

Now since $P(D) \in (0,1)$ and $D \subseteq D_{2\delta}^+$, r_1 is well defined and lies within $(0,1]$. Similarly, since $P(D^c) = 1 - P(D) \in (0,1)$ and $D^c \subseteq (D^c)_{2\delta}^+$, r_2 is well defined and also lies within $(0,1]$. This implies $\alpha r_1 \leqslant \alpha$ and $\beta r_2 \leqslant \beta$. Thus, the acceptance/rejection conditions $\mathbb{B}_{\bar{\mathbf{Z}}}(\bar{\mathbf{z}}, D_\delta^-) \geqslant \frac{1}{\beta \cdot r_2}$ and $\mathbb{B}_{\bar{\mathbf{Z}}}(\bar{\mathbf{z}}, D_\delta^+) \leqslant \alpha \cdot r_1$, using samples from $\bar{\mathbf{Z}}$, are stricter than those using $\bar{\mathbf{Y}}$. For acceptance, the region D is shrunk and Bayes' statistic is expected to be larger than a larger threshold $(1/\beta r_2 \geqslant 1/\beta)$ and for rejection, the region D is expanded and Bayes' statistic is expected to be smaller than a smaller threshold $(\alpha r_1 \leqslant \alpha)$.

Further, the region $D_\delta^+ \backslash D_\delta^-$ is an *indifference region* in the sense that, if $\bar{\mathbf{\Theta}}' \in D_\delta^+ \backslash D_\delta^-$, then Theorem 5 can neither accept nor reject H_0. This is why, some nested formulae cannot be verified by the approximate Bayes' test. Hence, we should only use the approximate Bayes' test if $\bar{\mathbf{\Theta}}'$ is not too close to the boundary of the region D. Otherwise, approximate Bayes' test will not pass the acceptance/rejection criterion. This problem arises when we verify nested HyperPCTL* formulae using approximate Bayes' test, but not when we verify non-nested formulae using the classical Bayes' test.

D. Hypothesis Testing by SPRT

Since we are comparing our approach to SPRT based SMC, we provide a short description of SPRT based hypothesis testing [10]. In SPRT, parameter $\bar{\mathbf{\Theta}}$ is assumed to be fixed and we decide between two most *indistinguishable hypotheses* instead of the original hypotheses (Equation 4). More precisely, we test

$$H_0' : \bar{\mathbf{\Theta}} \in D_\epsilon^- \text{ vs } H_1' : \bar{\mathbf{\Theta}} \in (D_\epsilon^+)^c,$$

for some small $\epsilon > 0$. A simpler statistic based on the log-likelihood function and Kullback-Leibler divergence is devised and H_0' or H_1' is accepted based on the position of the maximum likelihood estimate of $\bar{\mathbf{\Theta}}$ and the value of this statistic. Note that, $D_\epsilon^+ \backslash D_\epsilon^-$ is an *indifference region* in the sense that, if $\bar{\mathbf{\Theta}} \in D_\epsilon^+ \backslash D_\epsilon^-$, then we cannot test H_0 vs H_1 (Equation 4) using SPRT.

VII. STATISTICAL MODEL CHECKING

We will now discuss model checking of probabilistic hyperproperties using Bayes' test and approximate Bayes' test. Suppose a formula ψ, a DTMC $\mathcal{M}$ and a path assignment V is given. We refer to a HyperPCTL* formula as probabilistic if the top-level operator is $\mathbb{P}_D$. Let us note that, only two cases might arise for a probabilistic HyperPCTL* formula. On one hand, the formula can be non-nested, i.e., of the form $\mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \ldots, Pr^{\bar{\pi}_n}(\phi_n))$ where no ϕ_i contains a probabilistic subformula. On the other hand, a formula can be nested, i.e., of the form $\mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \ldots, Pr^{\bar{\pi}_n}(\phi_n))$ where some ϕ_i contains one or more probabilistic subformulae.

A. Verifying Non-nested Probabilistic Formula

As the base case, we will first discuss the verification of non-nested probabilistic formula on a DTMC. Let us assume, we have a formula ψ of the form $\mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \ldots, Pr^{\bar{\pi}_n}(\phi_n))$ where each ϕ_i is non-probabilistic (without $\mathbb{P}$ operator). We also assume each ϕ_i is closed by $\bar{\pi}_i$, i.e., $\Pi_{\phi_i} \subseteq \bar{\pi}_i$. Now let us define Bernoulli random variables $Y_{i=1,\ldots,n} : \text{Paths}(V, \bar{\pi}_i) \rightarrow \{0, 1\}$ as,

$$Y_i(\bar{\sigma}_i) = \begin{cases} 1 \text{ if } (\mathcal{M}, V[\bar{\pi}_i \rightarrow \bar{\sigma}_i]) \models \phi_i \\ 0 \text{ otherwise.} \end{cases}$$

Then $[\![Pr^{\bar{\pi}_i}(\phi_i)]\!]_V = P(Y_i = 1) = \Theta_i$, where Θ_i is the Bernoulli parameter of Y_i. Now our verification problem can be restated as a hypotheses testing problem,

$$H_0 : \bar{\mathbf{\Theta}} \in D \text{ vs } H_1 : \bar{\mathbf{\Theta}} \in D^c,$$

where $\bar{\mathbf{\Theta}} = (\Theta_1, \ldots, \Theta_n)$ is the parameter for the n dimensional random vector $\bar{\mathbf{Y}} = (Y_1, \ldots, Y_n)$. We say f is the prior for Θ_i if Θ_i is distributed with pdf f. In Bayesian framework, f assumed to be known. Assuming all Θ_i has the same prior f, we can easily compute the prior for $\bar{\mathbf{\Theta}}$, say $f_{\bar{\mathbf{\Theta}}}$, where $f_{\bar{\mathbf{\Theta}}}(\theta_1, \ldots, \theta_n) = \prod_{i=1}^n f(\theta_i)$. We can now apply Bayes' test repeatedly on larger and larger samples to deduce whether H_0 or H_1 holds with certain Type-I and Type-II error bounds. This procedure is described in the following algorithm.

Algorithm 1 BaseBayes: SMC of non-nested HyperPCTL* formula

Input: $\mathcal{M}$: DTMC, $\psi = \mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \ldots, Pr^{\bar{\pi}_n}(\phi_n))$: non-nested HyperPCTL* formula, V: path assignment, a, b: parameters for Beta prior, α, β: bounds on Type-I and Type-II errors

Output: Answer if $\mathcal{M}, V \models \psi$ with confidence α, β

1: $N \leftarrow 1$
2: **while** True **do**
3:     /*Generate data $\bar{\mathbf{y}} = (\mathbf{y_1}, \ldots, \mathbf{y_N})$ for $\bar{\mathbf{Y}}$*/
4:     **for** $k = 1$ to N **do**
5:         **for** $i = 1$ to n **do**
6:             Randomly sample $\bar{\sigma}_i$ from $\text{Paths}(V, \bar{\pi}_i)$
7:             **if** $(\mathcal{M}, V[\bar{\pi}_i \rightarrow \bar{\sigma}_i]) \models \phi_i$ **then**
8:                 $\mathbf{y_k}[i] \leftarrow 1$
9:             **else**
10:                $\mathbf{y_k}[i] \leftarrow 0$
11:             **end if**
12:         **end for**
13:     **end for**
14:     Calculate $\mathbb{B}_{\bar{\mathbf{Y}}}(\bar{\mathbf{y}}, D)$ using Equation 5
15:     **if** $\mathbb{B}_{\bar{\mathbf{Y}}}(\bar{\mathbf{y}}, D) \geqslant 1/\beta$ **then**
16:         Return True
17:     **else if** $\mathbb{B}_{\bar{\mathbf{Y}}}(\bar{\mathbf{y}}, D) \leqslant \alpha$ **then**
18:         Return False
19:     **else**
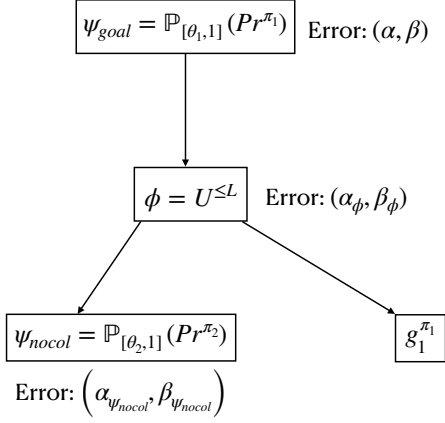20: $N \leftarrow 2 \cdot N$
21:     **end if**
22: **end while**

Fig. 2: A nested probabilistic formula

The correctness of Algorithm 1 follows directly from Theorem 3 (Bayes' test).

Theorem 7: [Correctness of BaseBayes] Let $\mathcal{M}$, ψ, V, a, b, α and β, be as in Algorithm 1. If $(\mathcal{M}, V) \models \psi$, then Algorithm 1 outputs True with probability at least $(1 - \alpha)$. If $(\mathcal{M}, V) \not\models \psi$, then Algorithm 1 outputs False with probability at least $(1 - \beta)$.

B. Verifying Nested Probabilistic Formula

We now describe our general SMC algorithm BayesSMC($\mathcal{M}, \psi, V, a, b, \alpha, \beta$) which can verify nested probabilistic formula on a DTMC. Here $\mathcal{M}$ is a DTMC, ψ is a (possibly) nested HyperPCTL* formula, V is a path assignment, a, b are parameters for the Beta prior and α, β are allowed upper bounds for Type-I and Type-II errors. Let us start with an overview of the algorithm.

1) Overview: Consider a nested probabilistic formula ψ of the form $\mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \ldots, Pr^{\bar{\pi}_n}(\phi_n))$, where ϕ_i has probabilistic subformulae ψ_{ij} for some $i \in [n]$. Given a DTMC $\mathcal{M}$ and a path assignment V, we want to verify if $(\mathcal{M}, V) \models \phi$ with Type-I and Type-II error bounds α, β respectively. We do this in a bottom up recursive approach where we first verify satisfaction of ψ_{ij} using BayesSMC with Type-I and Type-II error bounds α_{ij}, β_{ij} respectively. Then, we propagate the errors α_{ij}, β_{ij} using error propagation rules (Property 8) in order to calculate α_i, β_i, the Type-I and Type-II errors incurred by ϕ_i. Next, we calculate $\delta = \max_{i=1}^{n}\{\max\{\alpha_i, \beta_i\}\}$ from the errors incurred by the ϕ_i formulae. Finally, we apply the approximate Bayes' test (Theorem 5) using α, β and the derived δ to verify the satisfaction of ψ on $(\mathcal{M}, V)$.

Let us explain this using the nested formula ψ_{goal} (Equation 3). The formula tree for ψ_{goal} depicting its subformulae are shown in Figure 2. Suppose, we want to verify ψ_{goal} on a DTMC $\mathcal{M}$ and a path assignment V with error bounds α, β. We first verify the non-nested formula ψ_{nocol} with error bounds $\alpha_{\psi_{nocol}}, \beta_{\psi_{nocol}}$ on $(\mathcal{M}, V)$ using Bayes' test (Theorem 3). Then, we calculate α_ϕ, β_ϕ, Type-I and Type-II error bounds for the

subformula $\phi = (\psi_{nocol})U^{\leq K}(g_1^{\pi_1})$, using error propagation rules (Property 8). Clearly, $\delta = \max\{\alpha_\phi, \beta_\phi\}$ as ψ_{goal} has only one subformula ϕ. Finally, we verify ψ_{goal} on $(\mathcal{M}, V)$ using approximate Bayes' test (Theorem 5), with parameters α, β and δ.

2) Error Propagation: Let us define the recursive rules for error propagation now. Error is propagated in a bottom up manner, that is, from a subformula to its parent formula. For a formula ψ, let $E_1(\psi)$ denotes the Type-I error associated to ψ and $E_2(\psi)$ denotes the Type-II error associated to ψ. We now describe error propagation for a general (possibly nested) HyperPCTL* formula.

Let $\psi = \mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \ldots, Pr^{\bar{\pi}_n}(\phi_n))$ be a (possibly nested) HyperPCTL* formula, where each ϕ_i can have zero or more probabilistic subformula. Let us assume, by inductive hypothesis, $E_1(\phi_i)$ and $E_2(\phi_i)$ are Type-I and Type-II errors associated to ϕ_i. Let $\delta = \max_{i=1}^{n}\{\max\{E_1(\phi_i), E_2(\phi_i)\}\}$. Any $E_1(\phi_i)$ and $E_2(\phi_i)$ in $(0, 1)$ are allowed as long as $D_\delta^- \neq \varnothing$ (this is a necessary condition because to apply approximate Bayes' test, one needs to compute $\mathbb{B}_{\bar{\mathbf{Z}}}(\bar{\mathbf{z}}, D_\delta^-)$ for some random vector $\bar{\mathbf{Z}}$, and $\mathbb{B}_{\bar{\mathbf{Z}}}(\bar{\mathbf{z}}, D_\delta^-)$ is undefined in case $D_\delta^- = \varnothing$). Thus, we have a complete recursive definition of error propagation for a general HyperPCTL* formula given by the following property,

Property 8:
1) $E_1(a^\pi) = E_2(a^\pi) = 0$ for all $a \in AP$ and $\pi \in \Pi$;
2) $E_1(\neg\psi) = E_2(\psi)$, $E_2(\neg\psi) = E_1(\psi)$;
3) $E_1(\bigcirc\psi) = E_1(\psi)$, $E_2(\bigcirc\psi) = E_2(\psi)$;
4) $E_1(\psi_1 \wedge \psi_2) = E_1(\psi_1) + E_1(\psi_2)$, $E_2(\psi_1 \wedge \psi_2) = \max\{E_2(\psi_1), E_2(\psi_2)\}$;
5) $E_1(\psi_1 U^{\leq k}\psi_2) = k \cdot E_1(\psi_1) + E_1(\psi_2)$, $E_2(\psi_1 U^{\leq k}\psi_2) = (k + 1)\max\{E_2(\psi_1), E_2(\psi_2)\}$.
6) When ψ is nested, $E_1(\psi), E_2(\psi)$ can be any value in $(0, 1)$, whereas, δ is the maximum of all α_i, β_i, where α_i, β_i are error bounds for the subformulae ϕ_i.

Note that, rules 1-5 describe recursive error propagation for temporal formulae [2], whereas, rule 6 describes error propagation for (possibly nested) probabilistic formulae.

3) The Recursive Algorithm BayesSMC: Let $\psi = \mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \ldots, Pr^{\bar{\pi}_n}(\phi_n))$ be a nested formula where, each ϕ_i can consist of zero or more probabilistic subformulae. We define Bernoulli random variables Y_i as before. However, we cannot sample Y_i directly as we use SMC to check whether $(\mathcal{M}, V[\bar{\pi}_i \rightarrow \bar{\sigma}_i]) \models \phi_i$ which introduces uncertainty (since ϕ_i itself contains zero or more probabilistic subformulae). Thus, we sample from Bernoulli random variables $Z_{i=1,\ldots,n}$: Paths($V, \bar{\pi}_i$) $\rightarrow \{0, 1\}$ where,

$$Z_i(\bar{\sigma}) = \begin{cases} 1 & \text{if BayesSMC}(\mathcal{M}, \phi_i, V', a, b, \alpha_i', \beta_i') = \text{True} \\ 0 & \text{otherwise,} \end{cases}$$

and $V' = V[\bar{\pi}_i \rightarrow \bar{\sigma}_i]$. However, in this process we incur some errors with non-zero probability. More precisely, the Type-I error is given by $P(Z_i = 0 \mid (\mathcal{M}, V') \not\models \phi_i)$ and Type-II error is given by $P(Z_i = 1 \mid (\mathcal{M}, V') \models \phi_i)$.

Algorithm 2 BayesSMC: SMC of (possibly) nested HyperPCTL* formula

Input: $\mathcal{M}$: DTMC, $\psi = \mathbb{P}_D(Pr^{\bar{\pi}_1}(\phi_1), \ldots, Pr^{\bar{\pi}_n}(\phi_n))$: (possibly) nested HyperPCTL* formula, V: path assignment, a, b: parameters for Beta prior, α, β: bounds on Type-I and Type-II errors

Output: Answer if $\mathcal{M}, V \models \psi$ with confidence α, β
 1: **if** ψ is non-probabilistic **then**
 2:     Compute α, β from subformulae (Property 8)
 3:     **if** $(\mathcal{M}, V) \models \psi$ **then**
 4:         Return True
 5:     **else**
 6:         Return False
 7:     **end if**
 8: **end if**
 9: **if** ψ is non-nested **then**
10:     Return BaseBayes$(\mathcal{M}, \psi, V, a, b, \alpha, \beta)$
11: **end if**
12: $N \leftarrow 1$
13: **while** True **do**
14:     /*Generate data $\bar{z} = (\mathbf{z_1}, \ldots, \mathbf{z_N})$ for $\bar{Z}$*/
15:     **for** $i = 1$ to n **do**
16:         /*Recursively compute error bounds for ϕ_i
            by Property 8*/
17:         $\alpha'_i \leftarrow$ Type-I error bound of ϕ_i
18:         $\beta'_i \leftarrow$ Type-II error bound of ϕ_i
19:         **for** $k = 1$ to N **do**
20:             Randomly sample $\bar{\sigma}_i$ from Paths(V, π_i)
21:             $V' \leftarrow V[\bar{\pi}_i \rightarrow \bar{\sigma}_i]$
22:             **if** BayesSMC$(\mathcal{M}, \phi_i, V', a, b, \alpha'_i, \beta'_i)$ **then**
23:                 $\mathbf{z_k}[i] \leftarrow 1$
24:             **else**
25:                 $\mathbf{z_k}[i] \leftarrow 0$
26:             **end if**
27:         **end for**
28:     **end for**
29:     $\delta \leftarrow \max_{i=1}^n \{\max\{\alpha'_i, \beta'_i\}\}$
30:     Calculate constants r_1, r_2 from Theorem 5
31:     Calculate $\mathbb{B}_{\bar{z}}(\bar{z}, D_\delta^+)$ and $\mathbb{B}_{\bar{z}}(\bar{z}, D_\delta^-)$ (by Equation 5)
32:     **if** $\mathbb{B}_{\bar{z}}(\bar{z}, D_\delta^-) \geqslant 1/(\beta \cdot r_2)$ **then**
33:         Return True
34:     **else if** $\mathbb{B}_{\bar{z}}(\bar{z}, D_\delta^+) \leqslant \alpha \cdot r_1$ **then**
35:         Return False
36:     **else if** $\mathbb{B}_{\bar{z}}(\bar{z}, D_\delta^+) \geqslant 1/(\beta \cdot r_2)$ **then**
37:         **if** $\mathbb{B}_{\bar{z}}(\bar{z}, D_\delta^-) \leqslant \alpha \cdot r_1$ **then**
38:             Return Undecided
39:         **end if**
40:     **else**
41:         $N \leftarrow 2 \cdot N$
42:     **end if**
43: **end while**

Observe that, Type-I error is exactly equal to $P(Z_i = 0 \mid Y_i = 1)$ and Type-II error is exactly equal to $P(Z_i = 1 \mid Y_i = 0)$. Since by inductive hypothesis, Type-I and Type-II errors of BayesSMC$(\mathcal{M}, \phi_i, V', a, b, \alpha'_i, \beta'_i)$ are bounded by α'_i and

β'_i respectively, we have,

$$P(Z_i = 0 \mid Y_i = 1) \leqslant \alpha'_i \quad \text{and}$$
$$P(Z_i = 1 \mid Y_i = 0) \leqslant \beta'_i.$$

Thus we can say, the distributions of $\bar{\mathbf{Y}} = (Y_1, \ldots, Y_n)$ and $\bar{\mathbf{Z}} = (Z_1, \ldots, Z_n)$ are "close" by Equation 6. We can now apply Theorem 5 (approximate Bayes' test) to devise the recursive algorithm BayesSMC for verifying a (possibly) nested formula ψ on a DTMC $\mathcal{M}$.

The correctness of Algorithm 2 follows directly from Theorem 5 (approximate Bayes' test).

Theorem 9: [Correctness of BayesSMC] Let $\mathcal{M}$, ψ, V, a, b, α and β, be as in Algorithm 2. If $(\mathcal{M}, V) \models \psi$, then Algorithm 2 outputs True with probability at least $(1 - \alpha)$. If $(\mathcal{M}, V) \not\models \psi$, then Algorithm 2 outputs False with probability at least $(1 - \beta)$.

Comparison with SPRT based SMC: Let us compare our approach with SPRT based SMC [10]. In SPRT based SMC, for non-nested probabilistic formula, the verification problem is mapped to an equivalent n-dimensional hypothesis testing problem and solved by SPRT based hypothesis testing (Section VI-D); which is similar to our approach. On the other hand, for nested formula, SPRT based SMC uses verification results of subformulae directly while verifying the main formula. Thus, the total error incurred depends on the number of samples required to verify the main formula, which is not true for Bayesian SMC. Note that, both approaches provide Type-I and Type-II confidences for verification of probabilistic formulae.

VIII. EXPERIMENTAL EVALUATION

We evaluated our approach on the grid world robot navigation system discussed in section V. We consider $n \times n$ grids with two robots, for varying grid sizes n. The robots start from diagonally opposite cells, and their respective goals are to reach the horizontally opposite cells starting from their initial positions. We consider the collision avoidance property specified by the non-nested Formula ψ_{ca} (Equation 2) and the collision free goal reaching property for the first robot specified by the nested Formula ψ_{goal} (Equation 3).

We have implemented our algorithm in the Python tool box HyProVer. The recursive algorithm proceeds in a bottom-up fashion, where we need to check satisfiability of the subformulae and work our way up. However, we do not a priori know all the assignments on which the subformulae need to be evaluated, hence, a bottom-up approach would be expensive if we were to compute the satisfiability of the subformulae on all possible assignments. Instead, we have implemented an equivalent top-down algorithm where we start from the top and work our way down and evaluate the subformulae on only those assignments that are propagated down from the samples for the top-level formula. For the $\mathbb{P}$ operator, we only consider box constraints. Thus, integral can be evaluated on each dimension using the incomplete beta function and multiplied, to obtain the n-dimensional integrals required for calculating the Bayes' factor. We compare our Bayesian approach with our own implementation of the Frequentist approach based

on the SPRT method [10]. Note that, there are no publicly available probabilistic model checkers for HyperPCTL* for us to compare with.

Our verification results are summarized in Table II and Table III for Formula ψ_{ca} (Equation 2) and Formula ψ_{goal} (Equation 3), respectively, wherein we report the verification time in seconds and number of samples required for deduction of the satisfiability of the topmost probabilistic formula. Note that, the reported time and number of samples are the average values over multiple (50) runs of the same experiment. All experiments are performed on a machine having macOS Big Sur with Quad-Core Intel Core i7 2.8GHz× 1 Processor and 16GB RAM. We run each formula for at most 30 minutes and terminate the model-checker if it cannot provide a decision by that time.

Table I compares the Bayesian approach for different priors which are obtained by instantiating the Beta distribution parameters a and b. We used the uniform ($a = b = 1$), a left-skewed ($a = 5, b = 2$), a right-skewed ($a = 2, b = 5$) and a bell-shaped ($a = b = 2$) distribution as Beta priors. For $\alpha = \beta = 0.01$, we verified the nested Formula ψ_{goal} (Equation 3) describing collision free goal reaching for the first robot. We used different values of n, while keeping $K = 8$, $\theta_1 = 0.5$ and $\theta_2 = 0.5$ fixed. Also for $n = 4, 8$, we moved the goal of the first robot to grid position $(1, 1)$, so that the formula is satisfied. We observe that, the choice of prior affects the verification time and number of samples required by the topmost formula (averaged over 50 runs) in a minor manner. We use uniform prior for further comparison with SPRT, so that all parameter values are equally probable. Note that, a non-null indifference region always exists for SPRT (measured by the parameter ϵ) and like Bayesian, it also provides Type-I and Type-II guarantees for the correctness of a verification result [10].

The collision avoidance formula ψ_{ca} (Equation 2) depends on 3 parameters: n (grid size), K (bound for the until operator) and θ (probability threshold). In Table II, we varied n and K for different error bounds α, β and kept $\theta = 0.5$ fixed, as threshold value has little bearing on the verification time and required number of samples (from our observations as well as existing work on Bayesian SMC for Continuous Stochastic Logic [2]). We would like to note two main observations from Table II:

1) For non-nested formula, SPRT could not decide within time limit (30 minutes) whether collision probability lies below the threshold θ when collision probability was exactly 0. This is because we can only separate H_0 from H_1 using SPRT when $\bar{\Theta}$ does not lie in the *indifference region* (see [10]) and that is not true here. If the test region is $D = [0, \theta]$, then the indifference region, however small it might be, will always contain 0. Bayesian method does not have this problem, and it was able to provide inference in all the cases where SPRT failed. This shows a benefit of the Bayesian approach.

2) In those cases where both methods provided inference, Bayesian approach examined fewer samples and termi-

nated in shorter time in most of the cases as compared to the SPRT approach. This shows Bayesian approach is much more scalable than SPRT even for non-nested formulae. Note that, the inference provided by the two approaches always agree.

The collision free goal reaching formula for the first robot, ψ_{goal} (Equation 3), depends on 4 parameters: n (grid size), K (bound for until operator), θ_1 (probability threshold for the topmost probabilistic formula) and θ_2 (probability threshold for the probabilistic subformula ψ_{nocol}). In Table III, we varied n and K for different error bounds α, β and kept $\theta_1 = 0.3$ and $\theta_2 = 0.5$ fixed for the same reasons as mentioned before. Also for $n = 4, 8$, we moved the goal of the first robot to grid position $(0, 1)$, so that the formula is satisfied. We see that, the performance of SPRT is much worse for nested probabilistic formula as the verification never completes within the stipulated time limit (30 minutes) for any n, K and (α, β). The derogatory performance for the nested case is expected, since, the verification time and the corresponding number of samples grow exponentially with the nesting depth. Note that, SPRT was not tested on any nested HyperPCTL* formula in [10] as well. Thus, our evaluation demonstrates that Bayesian approach is superior to the Frequentist SMC for verification of hyperproperties and scales reasonably well for nested formulae as well.

IX. CONCLUSION

In this paper, we have developed a recursive statistical model checking algorithm for verifying discrete-time probabilistic hyperproperties on discrete-time Markov chains. Our broad approach consisted of mapping the HyperPCTL* verification problem to an n-dimensional hypotheses testing problem. We designed an algorithm based on random sampling followed by Bayes' test for non-nested HyperPCTL* formula, and extended it to the nested cases through a recursive algorithm that exploits an approximate Bayes' test. Finally, we used our algorithm to verify probabilistic hyperproperties like collision avoidance and collision free goal reaching on the grid world robot navigation scenarios. We compared the performance of our algorithm against the SPRT based algorithm discussed in [10] and showed that our algorithm performs better both in verification time and number of required samples for inference; a stark difference arises when we consider nested formulae.

For future work, we would like to develop a verification algorithm based on Bayes' test for hyperproperties over continuous time. Another interesting research direction would be to incorporate unbounded temporal (until) operators. This would enable the verification of unbounded probabilistic hyperproperties using light-weight methods such as Bayesian SMC.

	Uniform prior		Left-skewed prior		Right-skewed prior		Bell-shaped prior		
n	Samples	Time	Samples	Time	Samples	Time	Samples	Time	Status
4	40.64	0.893	63.04	1.371	46.08	0.985	62.08	1.342	TRUE
6	8.96	0.713	16.0	1.254	9.44	0.757	8.96	0.731	FALSE
8	121.6	8.933	207.68	15.401	124.8	9.354	170.24	12.573	TRUE
10	8.0	1.889	16.0	3.311	8.0	1.807	8.0	1.788	FALSE

TABLE I. Performance of HyProVer for different Beta priors

n	(α, β)	K	HyProVer		SPRT ($\epsilon = 0.01$)		SPRT ($\epsilon = 0.001$)		Status	
			Samples	Time	Samples	Time	Samples	Time	HyProVer	SPRT
4	0.01, 0.01	3	19.68	0.019	256.0	0.147	2048.0	0.994	TRUE	TRUE
6	0.01, 0.01	3	8.0	0.049	-	> 1800	-	> 1800	TRUE	UNDECIDED
8	0.01, 0.01	3	8.0	0.136	-	> 1800	-	> 1800	TRUE	UNDECIDED
10	0.01, 0.01	3	8.0	0.309	-	> 1800	-	> 1800	TRUE	UNDECIDED
4	0.001, 0.001	3	33.92	0.027	471.04	0.253	4096.0	1.979	TRUE	TRUE
6	0.001, 0.001	3	16.0	0.059	-	> 1800	-	> 1800	TRUE	UNDECIDED
8	0.001, 0.001	3	16.0	0.151	-	> 1800	-	> 1800	TRUE	UNDECIDED
10	0.001, 0.001	3	16.0	0.329	-	> 1800	-	> 1800	TRUE	UNDECIDED
4	0.01, 0.01	8	3817.6	3.270	3604.48	3.784	38666.24	33.261	FALSE	FALSE
6	0.01, 0.01	8	33.12	0.117	348.16	0.849	4096.0	8.887	TRUE	TRUE
8	0.01, 0.01	8	11.84	0.169	225.28	2.054	2048.0	9.131	TRUE	TRUE
10	0.01, 0.01	8	8.0	0.336	-	> 1800	-	> 1800	TRUE	UNDECIDED
4	0.001, 0.001	8	8785.92	7.452	6144.0	6.203	64880.64	55.780	FALSE	FALSE
6	0.001, 0.001	8	56.64	0.164	512.0	1.195	4096.0	8.889	TRUE	TRUE
8	0.001, 0.001	8	21.12	0.204	256.0	2.310	2048.0	8.916	TRUE	TRUE
10	0.001, 0.001	8	16.0	0.382	-	> 1800	-	> 1800	TRUE	UNDECIDED

TABLE II. SMC of collision avoidance formula

n	(α, β)	K	HyProVer		SPRT ($\epsilon = 0.01$)		SPRT ($\epsilon = 0.001$)		Status	
			Samples	Time	Samples	Time	Samples	Time	HyProVer	SPRT
4	0.01, 0.01	3	40.32	0.216	-	> 1800	-	> 1800	TRUE	UNDECIDED
6	0.01, 0.01	3	16.0	0.278	-	> 1800	-	> 1800	FALSE	UNDECIDED
8	0.01, 0.01	3	27.68	0.604	-	> 1800	-	> 1800	TRUE	UNDECIDED
10	0.01, 0.01	3	16.0	0.951	-	> 1800	-	> 1800	FALSE	UNDECIDED
4	0.001, 0.001	3	57.28	0.580	-	> 1800	-	> 1800	TRUE	UNDECIDED
6	0.001, 0.001	3	32.0	0.935	-	> 1800	-	> 1800	FALSE	UNDECIDED
8	0.001, 0.001	3	62.72	2.096	-	> 1800	-	> 1800	TRUE	UNDECIDED
10	0.001, 0.001	3	32.0	2.593	-	> 1800	-	> 1800	FALSE	UNDECIDED
4	0.01, 0.01	8	34.24	0.730	-	> 1800	-	> 1800	TRUE	UNDECIDED
6	0.01, 0.01	8	18.56	1.439	-	> 1800	-	> 1800	FALSE	UNDECIDED
8	0.01, 0.01	8	33.76	2.386	-	> 1800	-	> 1800	TRUE	UNDECIDED
10	0.01, 0.01	8	16.0	3.318	-	> 1800	-	> 1800	FALSE	UNDECIDED
4	0.001, 0.001	8	53.76	1.125	-	> 1800	-	> 1800	TRUE	UNDECIDED
6	0.001, 0.001	8	33.92	2.646	-	> 1800	-	> 1800	FALSE	UNDECIDED
8	0.001, 0.001	8	53.44	3.730	-	> 1800	-	> 1800	TRUE	UNDECIDED
10	0.001, 0.001	8	32.0	6.316	-	> 1800	-	> 1800	FALSE	UNDECIDED

TABLE III. SMC of collision free goal reaching formula for first robot

REFERENCES

[1] D. Sadigh, K. Driggs-Campbell, A. Puggelli, W. Li, V. Shia, R. Bajcsy, A. Sangiovanni-Vincentelli, S. S. Sastry, and S. Seshia, "Data-driven probabilistic modeling and verification of human driver behavior," in *2014 AAAI Spring Symposium Series*, 2014.

[2] R. Lal, W. Duan, and P. Prabhakar, "Bayesian statistical model checking for continuous stochastic logic," in *2020 18th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*. IEEE, 2020, pp. 1–11.

[3] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez, "Temporal logics for hyperproperties," in *International Conference on Principles of Security and Trust*. Springer, 2014, pp. 265–284.

[4] T.-H. Hsu, C. Sánchez, and B. Bonakdarpour, "Bounded model checking for hyperproperties," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2021, pp. 94–112.

[5] Y. Wang, S. Nalluri, and M. Pajic, "Hyperproperties for robotics: Planning via hyperltl," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8462–8468.

[6] S. Agrawal and B. Bonakdarpour, "Runtime verification of k-safety hyperproperties in hyperltl," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, 2016, pp. 239–252.

[7] M. R. Clarkson and F. B. Schneider, "Hyperproperties," *Journal of Computer Security*, vol. 18, no. 6, pp. 1157–1210, 2010.

[8] B. Finkbeiner, C. Hahn, M. Stenger, and L. Tentrup, "Monitoring hyperproperties," in *International Conference on Runtime Verification*. Springer, 2017, pp. 190–207.

[9] B. Finkbeiner and C. Hahn, "Deciding hyperproperties," *arXiv preprint arXiv:1606.07047*, 2016.

[10] Y. Wang, S. Nalluri, B. Bonakdarpour, and M. Pajic, "Statistical model checking for hyperproperties," in *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*. IEEE, 2021, pp. 1–16.

[11] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 286–296.

[12] J. W. Gray III, "Toward a mathematical foundation for information flow security," *Journal of Computer Security*, vol. 1, no. 3-4, pp. 255–294, 1992.

[13] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014.

[14] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *International conference on computer aided verification*. Springer, 2011, pp. 585–591.

[15] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 592–600.

[16] B. Finkbeiner, C. Hahn, and H. Torfah, "Model checking quantitative hyperproperties," in *International Conference on Computer Aided Verification*. Springer, 2018, pp. 144–163.

[17] Y. Wang, M. Zarei, B. Bonakdarpour, and M. Pajic, "Statistical verification of hyperproperties for cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–23, 2019.

[18] E. Clarke, A. Donzé, and A. Legay, "Statistical model checking of mixed-analog circuits with an application to a third order δ-σ modulator," in *Haifa Verification Conference*. Springer, 2008, pp. 149–163.

[19] R. Grosu and S. A. Smolka, "Monte carlo model checking," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2005, pp. 271–286.

[20] M. G. Merayo, I. Hwang, M. Núñez, and A. Cavalli, "A statistical approach to test stochastic and probabilistic systems," in *International Conference on Formal Engineering Methods*. Springer, 2009, pp. 186–205.

[21] D. E. Rabih and N. Pekergin, "Statistical model checking using perfect simulation," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2009, pp. 120–134.

[22] K. Sen, M. Viswanathan, and G. Agha, "Statistical model checking of black-box probabilistic systems," in *International Conference on Computer Aided Verification*. Springer, 2004, pp. 202–215.

[23] ——, "On statistical model checking of stochastic systems," in *International Conference on Computer Aided Verification*. Springer, 2005, pp. 266–280.

[24] H. L. Younes and R. G. Simmons, "Probabilistic verification of discrete event systems using acceptance sampling," in *International Conference on Computer Aided Verification*. Springer, 2002, pp. 223–235.

[25] S. Basu, A. P. Ghosh, and R. He, "Approximate model checking of pctl involving unbounded path properties," in *International Conference on Formal Engineering Methods*. Springer, 2009, pp. 326–346.

[26] Q. Cappart, C. Limbrée, P. Schaus, J. Quilbeuf, L.-M. Traonouez, and A. Legay, "Verification of interlocking systems using statistical model checking," in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. IEEE, 2017, pp. 61–68.

[27] D. Henriques, J. G. Martins, P. Zuliani, A. Platzer, and E. M. Clarke, "Statistical model checking for markov decision processes," in *2012 Ninth international conference on quantitative evaluation of systems*. IEEE, 2012, pp. 84–93.

[28] P. Zuliani, A. Platzer, and E. M. Clarke, "Bayesian statistical model checking with application to stateflow/simulink verification," *Formal Methods in System Design*, vol. 43, no. 2, pp. 338–367, 2013.

[29] E. M. Clarke, T. A. Henzinger, H. Veith, R. Bloem *et al.*, *Handbook of model checking*. Springer, 2018, vol. 10.

[30] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[31] M. Y. Vardi, "Automatic verification of probabilistic concurrent finite state programs," in *26th Annual Symposium on Foundations of Computer Science (SFCS 1985)*. IEEE, 1985, pp. 327–338.

[32] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini, "Approximate model checking of stochastic hybrid systems," *European Journal of Control*, vol. 16, no. 6, pp. 624–641, 2010.

[33] F. Ciesinski and M. Größer, "On probabilistic computation tree logic," in *Validation of Stochastic Systems*. Springer, 2004, pp. 147–188.

[34] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Verifying continuous time markov chains," in *International Conference on Computer Aided Verification*. Springer, 1996, pp. 269–276.

[35] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen, "Model-checking algorithms for continuous-time markov chains," *IEEE Transactions on software engineering*, vol. 29, no. 6, pp. 524–541, 2003.

[36] D. Bustan, S. Rubin, and M. Y. Vardi, "Verifying ω-regular properties of markov chains," in *International Conference on Computer Aided Verification*. Springer, 2004, pp. 189–201.

[37] H. Hermanns, B. Wachter, and L. Zhang, "Probabilistic cegar," in *International Conference on Computer Aided Verification*. Springer, 2008, pp. 162–175.

[38] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A bayesian approach to model checking biological systems," in *International conference on computational methods in systems biology*. Springer, 2009, pp. 218–234.

[39] J. Bogdoll, L. M. Ferrer Fioriti, A. Hartmanns, and H. Hermanns, "Partial order methods for statistical model checking and simulation," in *Formal Techniques for Distributed Systems*. Springer, 2011, pp. 59–74.

[40] K. G. Larsen and A. Skou, "Bisimulation through probabilistic testing," *Information and computation*, vol. 94, no. 1, pp. 1–28, 1991.

[41] G. Agha and K. Palmskog, "A survey of statistical model checking," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 28, no. 1, pp. 1–39, 2018.

[42] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *International conference on runtime verification*. Springer, 2010, pp. 122–135.

[43] B. Barbot, S. Haddad, and C. Picaronny, "Coupling and importance sampling for statistical model checking," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2012, pp. 331–346.

[44] S. Hadjis and S. Ermon, "Importance sampling over sets: A new probabilistic inference scheme." in *UAI*, 2015, pp. 355–364.

[45] A. Wald, "Sequential tests of statistical hypotheses," *The annals of mathematical statistics*, vol. 16, no. 2, pp. 117–186, 1945.

[46] S. Peyronnet, R. Lassaigne, and T. Herault, "Apmc 3.0: Approximate verification of discrete and continuous time markov chains," in *Third International Conference on the Quantitative Evaluation of Systems (QEST'06)*. IEEE, 2006, pp. 129–130.